Multiobjective Landscape Analysis and the Generalized Assignment Problem

Deon Garrett and Dipankar Dasgupta

University of Memphis, Memphis, TN, 38122, USA,
jdgarrtt@memphis.edu, dasgupta@memphis.edu

Abstract. The importance of tuning a search algorithm for the specific features of the target search space has been known for quite some time. However, when dealing with multiobjective problems, there are several twists on the conventional notions of fitness landscapes. Multiobjective optimization problems provide additional difficulties for those seeking to study the properties of the search space. However, the requirement of finding multiple candidate solutions to the problem also introduces new potentially exploitable structure. This paper provides a somewhat high-level overview of multiobjective search space and fitness landscape analysis and examines the impact of these features on the multiobjective generalized assignment problem.

1 Why Landscape Analysis

One of the foremost questions facing designers of metaheuristic algorithms for any sort of problem is how the structure of the objective function will affect the behavior of the search algorithm. It is known, and quite intuitive, that incorporating problem-specific knowledge into a search algorithm can often substantially increase the performance of the algorithm. However, given multiple conflicting options for building such algorithms, comparatively little is known concerning the right choices, or even the right information necessary to make good decisions.

This work examines a set of tools developed to help gain insights into how various algorithms navigate complex multiobjective search spaces. Many of these tools have previously been described in relation to conventional optimization problems. In such cases, the implications of extending the tools into the multiobjective realm are carefully examined. In this work, we propose some methods by which such relevant information may be obtained and exploited, and we apply these methods to a pair of classes of assignment problems exhibiting markedly different types of structure.

2 Fitness Landscape Analysis

As more and more researchers have turned their attention to modeling search algorithm performance, a number of techniques have been proposed to classify fitness landscapes, generally corresponding to fundamental properties of a given

search space. While most commonly defined in terms of classical single objective optimization, many of these properties have straightforward generalizations to the multiobjective domain. However, multiobjective optimization introduces additional features which may be analyzed and exploited by search algorithms. By studying these features, one should be able to design metaheuristic algorithms to better take advantage of the peculiarities of multiobjective optimization of a given problem.

Often, multiobjective search algorithms are defined in terms of a simpler, single objective algorithm. The n objectives of the original problem are scalarized using a particular weight vector, and the resulting single objective problem is attacked using the component search method. In these cases, the multiobjective problem may be completely characterized, for the purpose of modeling the performance of such an algorithm, by a family of related fitness landscapes. Each landscape is a window on the problem as viewed through a specific weight vector.

Abstractly, one may thus consider the ruggedness of adjacent landscapes. In fact, this notion of the similarity between nearby landscapes is what determines in part the success of different types of multiobjective local search algorithms. Following convention established by analysis of single objective landscapes, we may characterize a multiobjective problem as "smooth" if small changes to the underlying weight vector impose small changes on the fitness landscape. Conversely, a "rugged" multiobjective problem is one in which making a small change to the weight vector drastically alters the resulting fitness landscape.

Given a single good solution to a multiobjective optimization problem, the difficulty in finding other good solutions is largely determined by the smoothness of the family of landscapes. It is somewhat intuitive that smoothness implies that a good solution on a particular landscape should be nearby to good solutions on nearby landscapes. This spatial locality makes algorithms which attempt to build from one solution to a multiobjective problem to find many others more attractive. On the other hand, as the family of landscapes becomes more rugged, the information gained by finding one good solution becomes less useful in finding others.

The following sections describe a number of potentially useful metrics by which fitness landscapes, both single and multiobjective, may be characterized. In general, many of the tools in common use for analysis of single objective landscapes have fairly straightforward generalizations to the multiobjective realm. In addition, multiobjective algorithms which consist entirely of a sequence of independent runs of some underlying single objective optimization method may be directly studied using the single objective tools. However, it is also true that multiobjective optimization provides additional opportunities to exploit problem knowledge, and much of the goal of this work is to study these opportunities and apply the resulting knowledge to the problem of designing more effective algorithms. The remainder of this chapter is focused on defining a number of tools by which we may obtain such useful information about a given multiobjective problem instance.

2.1 Distribution of Local and Pareto Optima

Intuitively, the number and distribution of local optima would seem to have a profound impact on the performance of a general purpose search algorithm. A problem with but a single local optimum is by definition unimodal, and thus easily solved by any number of simple algorithms. As the number of local optima increase, the chances of becoming trapped in a local optimum are increased correspondingly. However, the distribution of local optima throughout the space is at least as important as the number of such optima. Classical notions such as deception are rooted entirely in the notion of unfortunate distributions of local optima.

One of the best-known examples of real-world problems with very different optima distributions is the comparison between the traveling salesman problem (TSP) and the quadratic assignment problem (QAP). In the TSP, problem instances exhibit what is commonly known as a "Big Valley" structure [1]. This term refers to the phenomena that almost all local optima are concentrated around a line that approaches the global optima, with the tour lengths of points along that line tending to increase as the distance to the global optimum increases.

In contrast, QAP instances tend to exhibit almost no structure when viewed in this same manner. The local optima for a typical QAP instance are very nearly uniformly distributed throughout the search space, with many, perhaps most, of all local optima lying close to the maximum possible distance from the global optimum. The difference in performance of a local search algorithm on TSP versus QAP is therefore quite dramatic.

From a multiobjective standpoint, many Pareto optimal solutions are also global optima of some single objective problem. Most commonly, given a Pareto front which is globally convex, there exists a weight vector for each Pareto optimal solution, a scalarization of the problem by which would result in the solution being the globally optimal solution of the resulting single objective problem. Thus, the distribution of local optima affects multiobjective problems just as it does for their single objective counterparts. However, multiobjective landscapes add an additional consideration, in that different Pareto optima are not generally local optima of the same single objective slice of the landscape. Therefore, the distribution of Pareto optima is in some sense a different aspect of the landscape than is the distribution of local optima leading to a single point on the Pareto front. In essense, you have to find both solutions, and the entire landscape can and often does change underneath you as you try to switch from one to the other.

2.2 Fitness Distance Correlation

Fitness distance correlation as a tool for modeling algorithm performance is based on the notion that good local optima should be near to the global optimum in terms of fitness as well. If this is the case, in principle there should be a clear trail from any local optimum to the global optimum in which each step requires

only small changes to the current solution. If instead, large jumps are required to move from a local optimum to a better solution nearer to the global optimum, most search algorithms may be expected to suffer.

Primarily defined in terms of single-objective optimization, fitness distance correlation is the correlation coefficient between the distance in objective space and the distance in parameter space between a set of randomly distributed local optima and the respective nearest global optimum to each. The standard Pearson correlation coefficient may be used to describe the results, although some useful information is not captured by this single summary statistic. Instead, scatter plots of parameter space distance versus objective space distance are often reported.

In the context of multiobjective optimization, the basic distinction is that the set of global optima is taken to be the set of nondominated solutions. While conceptually a simple extension, each Pareto optimal solution may or may not be the optimum of some mono-objective problem associated with a particular set of weights. The novelty in such a formulation is that, because each solution is, in essence, the optimum of a different fitness function, the correlations between nondominated solutions need bear no resemblance to the correlation between different *local* optima of a single function. Thus, considering the correlation between nondominated solutions can provide very useful information concerning the relative difficulty of moving "along" the Pareto front.

There are possibly other, more useful, ways to generalize this concept to multiobjective landscapes. The basic restriction of FDC is that one needs to get a single number indicating the distance to each optima. While one can certainly treat the Euclidean distance between a fitness vectors as the requisite metric, one may also consider, for example, the angle between vectors. If we treat distance as being defined by the angle between fitness vectors, then the nearest optimum will be, in a sense, "aligned" with the solution, in that they will have their component fitness values in the nearest proportion with one another. The nearest Pareto optimum under this definition will be that which lies adjacent on the Pareto front. The impact of these different choices on the resulting analysis is still an open question, but it is worthwhile to keep in mind the various possibilities that arise when dealing with vector valued optimization.

2.3 Ruggedness

Ruggedness is a somewhat vague notion, and a number of attempts have been made to formalize it [2]. However, intuitively, a landscape is rugged if there are many local optima of highly varying fitness concentrated in any constrained region of the space. Thus, it would seem that any definition of ruggedness, or conversely, smoothness must take into account both the number and distribution of local optima.

One straightforward measure of smoothness is to consider the correlation between adjacent points in the search space, where adjacency is dependent on the specification of a suitable neighborhood operator [3]. This correlation function value provides vital insights into the structure of a given search space under the

chosen operator. A high correlation coefficient implies that adjacent positions in the search space tend to have very similar fitness values. In this case, a local search algorithm might be expected to perform well, since it seems possible to exploit information gained by prior fitness function evaluations to effectively guide the choice of points to evaluate in the future.

As computing the true correlation requires exhaustive knowledge, researchers have often substituted autocorrelation instead. Autocorrelation typically arises in signal processing applications, and roughly speaking, measures the degree to which a time series is correlated with a time-shifted version of itself. In the current context of landscape analysis, one may construct a time series by setting out on a random walk and recording the fitness values at each point along the walk [3]. An uninformed random walk provides information about the overall ruggedness of the search space, which is certainly useful information for one seeking to design an effective algorithm.

However, if one knows the location of the global optima, or even of a set of high quality local optima, we can measure the autocorrelation of points along a path leading to these desirable solutions. For example, suppose we know the location of the global optimum for some problem. We can create a large number of random initial solutions, then send each one on a walk toward the global optimum. If the average autocorrelation is high, then we have reason to believe that other similar problem instances may be successfully attacked using fairly simple local improvement operators.

Multiobjective optimization provides another possible use for autocorrelation analysis. One vital piece of information concerns the relative difficulty of using previously located solutions to guide the search for additional Pareto optimal solutions versus performing a random restart to begin searching for other points. One way to attempt to answer this question is to look at the autocorrelation of random walks between known Pareto optimal solutions. If the path between Pareto optima is very rugged, it may be difficult for a two-phase algorithm to navigate the minefield of local optima. As a result, it may actually be beneficial to perform a restart to begin the search for additional Pareto optimal solutions.

2.4 Random Walk Analysis

Merz in [4] models crossover and mutation in hybrid evolutionary algorithms as random walks initiated from local optima. A series of mutations is represented by an undirected random walk starting from a local optimum. In contrast, crossover is modeled as a random walk starting at one local optimum and ending at another. Provided that the crossover operator is respectful [5, 4], this random walk between local optima explores the space in which offspring will be produced.

In [6], this method was extened to multiobjective optimization by considering the impact of crossover and mutation operating on the current nondominated frontier at various points during the evolutionary process. Using the mQAP as an example, it was shown that the ability of crossover to generate points closer to the Pareto front accounts for some of the success of a very good memetic

algorithm. On instances where crossover more closely resembled mutation, the memetic algorithm could not outperform a simpler local search metaheuristic.

Modeling genetic operators is not the only valid use of random walks, however. In the context of local search operators, random walks can help to provide estimates of many different properties of fitness landscapes. Watson et. al. [7, 8] used random walks to estimate the depth of each basin of attraction in job shop scheduling problems, and using the insights provided, developed an algorithm called I-JAR (iterated jump and re-descend) which was competitive with highly refined tabu search algorithms on JSP instances. I-JAR works by simply running a local search algorithm to a local optimum, then taking a random walk of length n, where n was empirically determined for the JSP, followed by an additional local improvement stage. The result is a very efficient algorithm which finds a local optimum, then "jumps" the minimum distance required to escape the basin of attraction before continuing the local search.

In multiobjective optimization, the notion of depth in relation to a basin of attraction is not so straightforward. Depending on both the location of the basin itself, or more accurately, the local optimum at the extremum of the basin, and the direction in which the search algorithm chooses to attempt its escape, the depth may vary dramatically. In general, even true Pareto optimal points will usually have a non-zero escape probability given an arbitrary direction of escape. This makes the notion of a barrier or basin more complicated in multiobjective landscapes, and further study is required to better understand the impact of changing direction during the search.

3 The Generalized Assignment Problem

The generalized assignment problem (GAP) deals with a set of m agents and a set of n tasks. Each task must be completed by exactly one agent. Each agent is allocated a specific number of resource units, and each agent requires a particular number of units to complete each task. Additionally, each agent incurs a specified cost for each task. The resource requirements and costs for a given task may differ between agents. The overall goal is to assign all tasks such that no agent violates the capacity constraints and the total costs incurred are minimized.

Formally, we may introduce an m-dimensional vector $\mathbf{B}$, with b_j denoting the total capacity alloted to agent j. We further introduce $m \times n$ matrices $\mathbf{A}$ and $\mathbf{C}$, denoting the resource matrix and the cost matrix respectively. Finally, we introduce an $m \times n$ binary matrix $\mathbf{X}$, with $x_{ij} = 1$ only if task i assigned to agent j by a particular candidate solution. The goal is thus to find such a solution so that

$$\min_{\mathbf{X}} \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} c_{ij}, \tag{1}$$

subject to

$$\sum_{i=1}^{m} x_{ij} a_{ij} \leq b_j \quad \forall\, j : 1 \leq j \leq n \tag{2}$$

and

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall\, i : 1 \leq i \leq m. \tag{3}$$

(2) are known as the *capacity constraints*, and (3) are called the *semi-assignment constraints*.

The GAP is known to be $\mathcal{NP}$-hard [9], and exact algorithms have proven tractable only for problems of a few hundred tasks or less [10]. Thus, for large instances, heuristic and metaheuristic methods have received a great deal of attention, including tabu search approaches [11, 12], variable depth search [13, 14], ant colony optimization [15], evolutionary algorithms [16], and more recently, path relinking algorithms [17–20].

Most experimental studies of the GAP have chosen a benchmark set of randomly generated instances divided into five classes, imaginatively named A, B, C, D, and E. The A and B-type instances are not considered challenging to modern algorithms and are not considered in this work. The D and E-type instances considered more difficult than the C-type due to the fact that the costs are inversely related to the resource allocations in the former instances.

Type C:
$$\begin{aligned} a_{ij} &= \mathrm{U}(5, 25) \\ c_{ij} &= \mathrm{U}(10, 50) \\ b_i &= 0.8 \textstyle\sum_j \frac{a_{ij}}{m} \end{aligned}$$

Type D:
$$\begin{aligned} a_{ij} &= \mathrm{U}(1, 100) \\ c_{ij} &= 111 - a_{ij} + \mathrm{U}(-10, 10) \\ b_i &= 0.8 \textstyle\sum_j \frac{a_{ij}}{m} \end{aligned}$$

Type E:
$$\begin{aligned} a_{ij} &= 1 - 10 \ln(\mathrm{U}(0, 1]) \\ c_{ij} &= \tfrac{1000}{a_{ij}} - 10\mathrm{U}[0, 1] \\ b_i &= 0.8 \textstyle\sum_j \frac{a_{ij}}{m} \end{aligned}$$

3.1 The Multiobjective GAP

Like many other combinatorial optimization problems, the GAP is often applicable in situations calling for simultaneous optimization of more than one objective function. We propose a straightforward extension of the GAP to a multiobjective problem. The reasons for this are to provide an additional test for multiobjective algorithms, but also because we feel that a multiobjective benchmark problem featuring constraints and a quite different type of structure to the now well-known mQAP should be added to the suite of widely available benchmarks. In the remainder of this paper, the mGAP will be described and examined with particular attention to the structure of the resulting search space and its impact on MOGLS algorithm performance.

The most basic formulation of the mGAP is to augment the description in the previous section with additional cost matrices. Each task must then be assigned

to a single agent in such a way as to minimize a vector of costs. Formally, the optimization problem is thus to find a solution s such that

$$\min_{s \in \Lambda} \sum_{i=1}^{M} C^k_{s_i,i}, \tag{4}$$

where C^k denotes the k^{th} cost matrix and the min operation denotes vector minimization, or Pareto optimization. The constraints are unchanged from the single objective version of the problem.

Proceeding analogously to Knowles and Corne for the mQAP [21, 22], the correlation between cost matrices would seem to be an important indicator of search algorithm performance. The analysis presented for the mQAP indicates that in some situations, a MOSA-like strategy could be quite effective. In such an algorithm, a single solution is optimized with respect to some weight vector. The weights are then modified slightly and a new optimization pass is initialized from the locally optimal solution found in the first stage. The process could continue until some termination criteria is satisfied, at which point the algorithm could terminate or choose to restart from a new randomly chosen starting point and a new weight vector.

However, unlike the mQAP, the mGAP is constrained. In the mGAP, the capacity constraints make it unlikely that large numbers of neighboring solutions are feasible. One may allow the search to visit infeasible regions of the space, provided some appropriate method were in place to guide the search back toward good feasible solutions. Alternately, the algorithm may be prohibited from exploration outside the feasible region. In the latter case, "neighbors" along the Pareto optimal front may be disconnected. This could have profound implications on the ability of the two-phase local search algorithms to adequately explore the Pareto optimal front.

A related issue in the description and formulation of the mGAP is the tightness of the constraints. Intuitively, if the given capacity constraints for each agent roughly equal the average resource usage times the average number of tasks per agent, then the problem would seem to be quite difficult. Simply finding a feasible solution may be very challenging.

Furthermore, the relationship between the resource usage and costs can also impact the performance of search algorithms. If the costs are inversely related to the resource usage, then the most cost effective solutions are likely to involve mappings of tasks to agents that incur large resource usage, thus moving ever nearer to the boundaries of the feasible regions of the space. This can be of particular importance to neighborhood-based search algorithms such as those considered in this work. Additionally, two of the three classes of random instances involved cost matrices which are negatively correlated with the resource matrices. As we have previously seen, correlation between cost matrices may be relevant in determining search algorithm performance. If, for example, one generates a mGAP instance with negatively correlated cost matrices, one of the objectives will then be positively correlated with the resource matrix. It might

thus be worth exploring additional forms of random instances in the case of multiobjective optimization.

However, in generalizing the GAP to include multiple objectives, additional problems arise. Standard problem instances are designed to ensure that the feasible region is somewhat tight and the type D and E instances are designed to be more difficult by correlating their cost function with the resource requirements. Generating multiple cost matrices based on these definitions yields problem instances in which essentially all Pareto optimal solutions lie in a very small feasible region, as the high correlation (or anti-correlation) means that the resulting costs matrices are very similar. Care must be taken in generating instances so that the resulting problem is not degenerate in some form.

3.2 Local Search Algorithms for the GAP

The GAP, and by extension, the mGAP admits more than one possible type of move in relation to a local search algorithm. Unlike the QAP, in which the swap is essentially universally used as the basis of local search methods, local search algorithms for the GAP tend to utilize both *shifts* and swaps. In a shift operation, a single task is reassigned to a different agent. Shifts can be effective when trying to move from an infeasible to a feasible solution, since they can take free up resources from one agent without adding other resources in return. Of course, if a solution is feasible, a shift may be unlikely to maintain feasibility for exactly the same reason.

Unlike shifts, swaps cannot alter the number of tasks assigned to any agents. This lack of explorative power would likely yield very poor performance. Therefore, most local search approaches to the GAP rely on some combination of shifts and swaps. In principle, the swap neighborhood is a strict subset of the neighborhood imposed by two subsequent shifts, and therefore might be viewed as extraneous. However, in practice, the ability to perform a true swap is useful enough to warrant special support.

There exist different approaches to utilizing shifts and swaps in the GAP neighborhood. Possibly the simplest effective method is to randomly select either a shift or swap at each iteration. That approach will be adopted here, largely for two reasons. First, a local search algorithm using such a neighborhood can be reasonably effective. However, the major reason is one of convenience. In order to gather information concerning the properties of the fitness landscape, it is necessary to be able to compute the number of moves which separate any two solutions. Using a full shift/swap neighborhood, it is relatively easy to compute the number of moves required to transform a solution into any other feasible solution. By contrast, in one of the most recent approaches [10], ejection chains are used to search the GAP space. Ejection chains are formed by a series of n successive shift operations. Because the neighborhood size grows exponentially with n, that work utilizes classical integer programming bounds to prune the neighborhood to a manageable number of solutions. While providing high performance,

the highly dynamic neighborhoods prohibit the sort of analysis required by this work.

To compute the distance between any two candidate solutions under the shift/swap neighborhood, let us consider two candidate solutions A and B. Let A be considered the reference solution, and we want to compute the minimum number of moves necessary to reach A starting from B. Each locus of the two solutions is compared, and where they assign a given task to different agents, a move is deemed necesssary. To determine what type of move should be taken, a linear search of the remaining unexamined positions of B is performed, looking for a task in B assigned to the agent assigned to the current task in A. If such a task is found in B, then a swap of the two positions in B moves B two steps closer to A and is thus taken. If no such element is found, then any swap in B must by necessity move just one step closer, and thus is no more effective than a shift. When B is equal to A, the number of moves taken denotes the distance between the original two solutions.

3.3 Examining the mGAP Pareto Front

To gain some initial insights into the structure of mGAP instances, several sets of randomly generated toy problems were considered. The problem sizes were deliberately kept small so that the search spaces could be exhaustively searched in reasonable periods of time. Each problem was then enumerated and all Pareto optimal points recorded. Table 1 shows the number and types of instances examined along with some summary statistics on the resulting Pareto optimal sets.

In Table 1, Cor refers to the coefficient of correlation between the two cost matrices, $|PF|$ to the number of Pareto optimal solutions, and r to the Pearson coefficient of fitness-distance correlation. Two important conclusions can be drawn immediately from inspection of the Pareto fronts on these small problems. First, the FDC coefficient (denoted by r in the table) is much higher than is observed in the mQAP, where it is typically very close to zero. This implies that there should exhibit more structure among Pareto optimal solutions than we typically see in mQAP instances. This is promising for algorithms which attempt to utilize information from previously located efficient solutions to help generate additional efficient solutions.

Secondly, E-type problems with uncorrelated cost matrices seem to admit very few Pareto optimal solutions, often only one. Extrapolating to larger problems, the Pareto fronts for such problems would seem to be restricted to a very small region of the search space. However, with respect to this property, such extrapolation is fraught with danger. It seems equally likely that the underlying cause of this behavior is an increase in the tightness of the constraints. If this is true, then the trend toward small numbers of Pareto optimal solutions might continue as the problem size was increased. However, the actual solutions might be somewhat more distributed throughout the space, with large regions of infeasibility in between. More research is required to answer this question with any certainty.

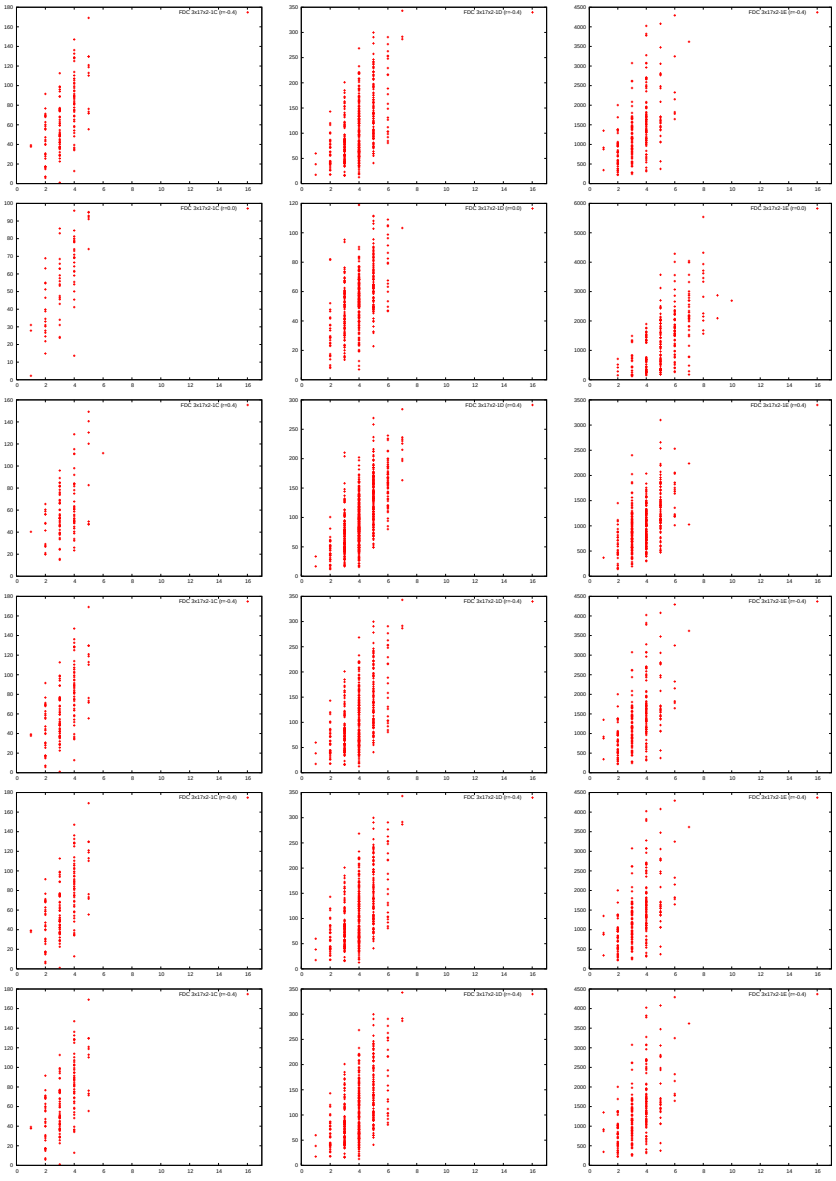

Fig. 1. Fitness distance correlation plots for the various instances considered. Columns left to right are $r = \{$`C, D, E`$\}$ type instances. Row 1, 3x17 $r = -0.4$; Row 2, 3x17 $r = 0.0$; Row 3, 3x17 $r = 0.4$; Row 4, 5x12 $r = -0.4$; Row 5, 5x12 $r = 0.0$; Row 6, 5x12 $r = 0.4$.

Table 1. Toy mGAP instances examined. For each problem type, number of objectives, and correlation between cost matrices, the number of Pareto optimal solutions, the fitness distance correlation coefficient, and the ratio of infeasible to feasible solutions along each path to the front were recorded.

Size	Type	Obj	Cor	$\|PF\|$	Pearson's r	$frac_{infeas}$
3x17	C	2	-0.4	14.43±4.83	0.48±0.11	0.86
3x17	D	2	-0.4	13.67±4.56	0.45±0.12	0.86
3x17	E	2	-0.4	17.90±8.19	0.59±0.09	0.82
3x17	C	2	0.0	9.23±5.39	0.53±0.13	0.88
3x17	D	2	0.0	7.16±2.29	0.44±0.08	0.87
3x17	E	2	0.0	1.13±0.43	0.52±0.09	0.88
3x17	C	2	0.4	7.30±3.27	0.48±0.28	0.89
3x17	D	2	0.4	9.73±3.93	0.44±0.10	0.87
3x17	E	2	0.4	8.80±4.43	0.66±0.08	0.83
5x12	C	2	-0.4	14.30±7.05	0.44±0.15	0.89
5x12	D	2	-0.4	12.27±5.00	0.46±0.11	0.85
5x12	E	2	-0.4	16.53±6.13	0.46±0.11	0.82
5x12	C	2	0.0	9.37±4.12	0.51±0.15	0.91
5x12	D	2	0.0	5.50±2.21	0.40±0.09	0.91
5x12	E	2	0.0	1.20±0.41	0.41±0.18	0.92
5x12	C	2	0.4	6.17±2.57	0.56±0.12	0.91
5x12	D	2	0.4	9.13±2.71	0.46±0.11	0.87
5x12	E	2	0.4	7.73±3.04	0.52±0.12	0.83

A consequence of the positive fitness distance correlation coefficient is that there exist local optima at many different distances to the Pareto front. In contrast, in mQAP instances, it is usually the case that all local optima are fairly uniformly distributed with no two lying too close together. Although this structure in mGAP instances can be usefully exploited by search algorithms attempting to move along the Pareto front, the presence of constraints means that not all solutions on a path between two efficient solutions need be feasible. If these intermediate solutions lie deep within infeasible regions of the space, an algorithm might need to take special measures to allow large constraint violations to occur during the search.

To test the extent to which constraint violations occur between efficient solutions, the number of infeasible solutions visited along the shortest path between each local optima and the nearest efficient solution was recorded during the FDC analysis described above. The percentage of the path to the nearest Pareto optimal solution is shown in Table 1 as $frac_{infeas}$. Across all types and sizes of instances, this percentage remains reasonably stable between about 82 and 92 percent. Looking at the FDC scatter plots shown in Figure 1, it is clear that while many of the local optima are a significant distance from the Pareto front, there is a moderately strong positive correlation and reasonably significant structure in the landscape. Despite this structure, the high percentage of infeasible solutions along the shortest path to the front implies that an algorithm seeking

Table 2. Performance of the MOTS and Two-Phase TS algorithms on a set of $20x100$ mGAP instances, measured as the hypervolume mean and standard deviation. All results are averaged over 30 trials.

Instance	MOTS	Two-Phase TS
1	1.5562e7 ± 2.26e5	1.6415e7 ± 1.91e5
2	1.3271e7 ± 1.71e5	1.3994e7 ± 1.70e5
3	1.5562e7 ± 2.26e5	1.6415e7 ± 1.91e5
4	1.3271e7 ± 1.71e5	1.3994e7 ± 1.70e5
5	1.5562e7 ± 2.26e5	1.6415e7 ± 1.91e5
6	1.5562e7 ± 2.26e5	1.6415e7 ± 1.91e5
7	1.3271e7 ± 1.71e5	1.3994e7 ± 1.70e5
8	1.5562e7 ± 2.26e5	1.6415e7 ± 1.91e5
9	1.3271e7 ± 1.71e5	1.3994e7 ± 1.70e5
10	1.5562e7 ± 2.26e5	1.6415e7 ± 1.91e5

to approach the Pareto front from a random local optimum may need to move deep into the infeasible regions of the space.

In addition to the toy instances, a set of ten $20x100$ bi-objective GAP instances were also generated, and optimized using both a straightforward multiobjective tabu search algorithm as well as a two-phase tabu search method. The structure exhibited by the GAP would seem to provide the two-phase algorithm the ability to better exploit its prior work than would the much more random structure inherent in QAP instances, for example. Figure 2 shows the median empirical attainment surfaces over 30 trials of the two algorithms applied to one of the larger instances. The results are consistent across the ten instances, with the two phase algorithm showing a moderate, but consistent and statistically significant increase in performance, as shown in Table 2.

However, as shown in Table 1, a high percentage of the points leading to a new Pareto optimal solution are generally infeasible. This can adversely affect the ability of a simple two-phase local search algorithm to move along the Pareto front, once it has located a single solution. In multiobjective optimization, where it is important to be able to move between good nondominated solutions during the search, the effects of tight constraints are not yet well understood. An algorithm that could selectively and dynamically adjust the degree to which it allowed constraint violations might allow even greater improvements in multiobjective optimization via these types of two-phase heuristics.

4 Conclusions

It is known that no single algorithm is appropriate for all search and optimization tasks. Taking advantage of domain specific knowledge and/or known structural properties of a given search space is a must in designing a high-performance algorithm. However, in the multiobjective realm, comparatively little attention has been paid to the unique aspects of multiobjective landscapes. In particular,

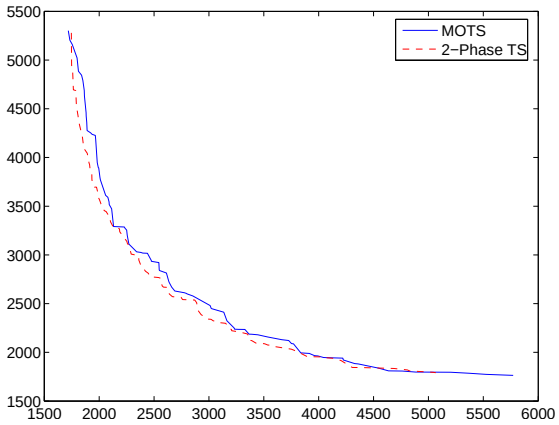

Fig. 2. Empirical attainment surfaces for a simple multiobjective tabu search and a simple two-phase variant for a sample 20x100 bi-objective generalized assignment problem instance.

not all multiobjective algorithms approach the Pareto front via a single synthesized objective function which can be studied using existing landscape analysis tools. Many multiobjective evolutionary algorithms in particular fit this description, instead relying on a type of coevolution to move the population as a whole toward the Pareto front.

This paper has described some basic tools for examining the properties of multiobjective fitness landscapes. Many of these tools are straightforward generalizations of well-known ideas from the world of single-objective algorithms. However, the unique requirements of multiobjective optimization mean that there are additional types of information that may be exploited by effective algorithms, and there can be subtle differences in the interpretation of familiar techniques. Only by gaining a further understanding of multiobjective search spaces can we hope to systematically design faster and more effective algorithms.

References

1. Boese, K.D.: Cost versus distance in the traveling salesman problem. Technical Report TR-950018, University of California at Los Angeles (1995)
2. Hoos, H., Stützle, T.: Stochastic Local Search: Foundations and Applications. Elsevier (2005)
3. Weinberger, E.D.: Correlated and uncorrelated fitness landscapes and how to tell the difference. Biological Cybernetics **63**(5) (1990) 325–336
4. Merz, P.: Advanced fitness landscape analysis and the performance of memetic algorithms. Evolutionary Computation **12**(3) (2004) 303–325
5. Radcliffe, N.: Forma analysis and random respectful recombination. In Belew, R., Booker, L., eds.: Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufman (1991)

6. Garrett, J.D., Dasgupta, D.: Analyzing the performance of hybrid evolutionary algorithms on the multiobjective quadratic assignment problem. In: Proceedings of the IEEE Congress on Evolutionary Computation. (2006)
7. Watson, J.P.: Empirical Modeling and Analysis of Local Search Algorithms for the Job-Shop Scheduling Problem. PhD thesis, Colorado State University, Fort Collins, CO (2003)
8. Watson, J.P., Howe, A., Whitley, L.D.: An analysis of iterated local search for job-shop scheduling. In: Proceedings of the Fifth Metaheuristics International Conference. (2003)
9. Sahni, S., Gonzalez, T.: P-complete approximation problems. Journal of the ACM **23**(3) (1976) 555–565
10. Yagiura, M., Ibaraki, T., Glover, F.: An ejection chain approach for the generalized assignment problem. INFORMS Journal on Computing **16**(2) (2004) 133–151
11. Díaz, J.A., Fernández, E.: A tabu search heuristic for the generalized assignment problem. European Journal of Operational Research **132**(1) (2001) 22–38
12. Laguna, M., Kelly, J.P., González-Velarde, J.L., Glover, F.: Tabu search for the multilevel generalized assignment problem. European Journal of Operational Research **82**(1) (1995)
13. Yagiura, M., Yamaguchi, T., Ibaraki, T.: A variable depth search algorithm with branching search for the generalized assignment problem. Optimization Methods and Software **10**(3) (1998) 419–441
14. Racer, M., Amini, M.M.: A robust heuristic for the generalized assignment problem. Annals of Operations Research **50**(1) (1994) 487–503
15. Lourenço, H.R., Serra, D.: Adaptive search heuristics for the generalized assignment problem. Mathware and Soft Computing **9**(3) (2002) 209–234
16. Chu, P.C., Beasley, J.E.: A Genetic Algorithm for the Generalized Assignment Problem. Computers and Operations Research **24**(1) (1997) 17–23
17. Alfandari, L., Plateau, A., Tolla, P.: A two-phase path relinking algorithm for the generalized assignment problem. In: Proceedings of the Fourth Metaheuristics International Conference. (2001) 175–179
18. Alfandari, L., Plateau, A., Tolla, P.: A path relinking algorithm for the generalized assignment problem. In Resende, M.G.C., Sousa, J.P., eds.: Metaheuristics: Computer Decision-Making. Kluwer Academic Publishers (2004) 1–17
19. Yagiura, M., Ibaraki, T., Glover, F.: An effective metaheuristic algorithm for the generalized assignment problem. In: 2001 IEEE International Conference on Systems, Man, and Cybernetics. (2001) 242–250
20. Yagiura, M., Ibaraki, T., Glover, F.: A path relinking approach for the generalized assignment problem. In: Proceedings of the International Symposium on Scheduling. (2002) 105–108
21. Knowles, J., Corne, D.: Towards landscape analyses to inform the design of a hybrid local search for the multiobjective quadratic assignment problem. In Abraham, A., del Solar, J.R., Koppen, M., eds.: Soft Computing Systems: Design, Management and Applications. IOS Press (2002) 271–279
22. Knowles, J., Corne, D.: Instance generators and test suites for the multiobjective quadratic assignment problem. In: Evolutionary Multi-Criterion Optimization (EMO 2003), Second International Conference, Faro, Portugal (April 2003) 295–310